

RRRRRRRRRRRR		UUU	UUU	NNN	NNN	000000000	FFFFFFFFFFF	FFFFFFFFFFF
RRRRRRRRRRRR		UUU	UUU	NNN	NNN	000000000	FFFFFFFFFFF	FFFFFFFFFFF
RRRRRRRRRRRR		UUU	UUU	NNN	NNN	000000000	FFFFFFFFFFF	FFFFFFFFFFF
RRR	RRR	UUU	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNNNNN	NNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNNNNN	NNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNNNNN	NNN	000	FFF	FFF
RRRRRRRRRRRR		UUU	UUU	NNN	NNN	000	FFFFFFFFF	FFFFFFFFF
RRRRRRRRRRRR		UUU	UUU	NNN	NNN	000	FFFFFFFFF	FFFFFFFFF
RRRRRRRRRRRR		UUU	UUU	NNN	NNN	000	FFFFFFFFF	FFFFFFFFF
RRR	RRR	UUU	UUU	NNN	NNNNNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNNNNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNNNNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUUUUUUUUUUUUUUU	NNN	NNN	000000000	FFF	FFF	
RRR	RRR	UUUUUUUUUUUUUUUU	NNN	NNN	000000000	FFF	FFF	
RRR	RRR	UUUUUUUUUUUUUUUU	NNN	NNN	000000000	FFF	FFF	

```
NN      NN  DDDDDDDD  XX      XX  TTTTTTTTTT  EEEEEEEEEEE  XX      XX
NN      NN  DDDDDDDD  XX      XX  TTTTTTTTTT  EEEEEEEEEEE  XX      XX
NN      NN  DD        DD  XX      XX  TT          EE          XX      XX
NN      NN  DD        DD  XX      XX  TT          EE          XX      XX
NNNN     NN  DD        DD  XX      XX  TT          EE          XX      XX
NNNN     NN  DD        DD  XX      XX  TT          EE          XX      XX
NN  NN    NN  DD        DD  XX      XX  TT          EE          XX      XX
NN  NN    NN  DD        DD  XX      XX  TT          EE          XX      XX
NN      NN  DD        DD  XX      XX  TT          EE          XX      XX
NN      NN  DD        DD  XX      XX  TT          EE          XX      XX
NN      NN  DD        DD  XX      XX  TT          EE          XX      XX
NN      NN  DDDDDDDD  XX      XX  TT          EEEEEEEEEEE  XX      XX
NN      NN  DDDDDDDD  XX      XX  TT          EEEEEEEEEEE  XX      XX
```

```
LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS
```



```
0001 0 %TITLE 'NDXTEx -- RUNOFF to TEX conversion'
0002 0 MODULE NDXTEx (IDENT = 'v04-000'
0003 0 %BLISS32 [, ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE, NONEXTERNAL = LONG_RELATIVE)]
0004 0 ) =
0005 1 BEGIN
0006 1
0007 1
0008 1 *****
0009 1 *
0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0012 1 * ALL RIGHTS RESERVED.
0013 1 *
0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0019 1 * TRANSFERRED.
0020 1 *
0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0023 1 * CORPORATION.
0024 1 *
0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0027 1 *
0028 1 *
0029 1 *****
0030 1
0031 1
0032 1 ++
0033 1 FACILITY:
0034 1 DSR (Digital Standard RUNOFF) /DSRPLUS DSRINDEX/INDEX Utility
0035 1
0036 1 ABSTRACT:
0037 1 This module contains code to convert RUNOFF input text lines to
0038 1 TEX text lines.
0039 1
0040 1 ENVIRONMENT: Transportable
0041 1
0042 1 AUTHOR: JPK
0043 1
0044 1 CREATION DATE: March 1983
0045 1
0046 1 MODIFIED BY:
0047 1
0048 1 --
```

```
: 50      0049 1 |
: 51      0050 1 | TABLE OF CONTENTS:
: 52      0051 1 |
: 53      0052 1 |
: 54      0053 1 | FORWARD ROUTINE
: 55      0054 1 |   RNOTEX : NOVALUE;
: 56      0055 1 |
: 57      0056 1 |
: 58      0057 1 |   INCLUDE FILES:
: 59      0058 1 |
: 60      0059 1 |
: 61      0060 1 | LIBRARY 'NXPORT:XPORT';
: 62      0061 1 |
: 63      L 0062 1 | %IF %BLISS (BLISS32)
: 64      0063 1 | %THEN
: 65      0064 1 |
: 66      0065 1 | REQUIRE 'REQ:NDXVMSREQ';
: 67      0346 1 |
: 68      0347 1 | %FI
: 69      0348 1 |
: 70      0349 1 |
: 71      0350 1 |   MACROS:
: 72      0351 1 |
: 73      0352 1 |
: 74      0353 1 |   EQUATED SYMBOLS:
: 75      0354 1 |
: 76      0355 1 |
: 77      0356 1 | LITERAL
: 78      0357 1 |   TRUE = 1,
: 79      0358 1 |   FALSE = 0,
: 80      0359 1 |   BUFFER_SIZE = 2000;
: 81      0360 1 |
: 82      0361 1 | LITERAL
: 83      0362 1 |   NORMAL_FONT = 0,
: 84      0363 1 |   BOLD_FONT = 1,
: 85      0364 1 |   ITALIC_FONT = 2,
: 86      0365 1 |   BOLD_ITALIC = 3;
: 87      0366 1 |
: 88      0367 1 |
: 89      0368 1 |   OWN STORAGE:
: 90      0369 1 |
: 91      0370 1 |
: 92      0371 1 |   EXTERNAL REFERENCES:
: 93      0372 1 |
: 94      0373 1 | EXTERNAL LITERAL
: 95      0374 1 |   TAB : UNSIGNED (8);
```

! Generate TEX output

```
! Names for various fonts.
! NOTE: code in RNOTES makes use of the
! fact that NORMAL_FONT is zero and that
! (BOLD_FONT OR ITALIC_FONT) is equal to
! BOLD_ITALIC.
```



```

: 97 0375 1 %SBTTL 'RNOTEX - Generate TEX Output from RUNOFF text'
: 98 0376 1 GLOBAL ROUTINE RNOTEX (RNO_LEN, RNO_PTR, DSC) : NOVALUE =
: 99 0377 1 ++
: 100 0378 1
: 101 0379 1 FUNCTIONAL DESCRIPTION:
: 102 0380 1
: 103 0381 1 This routine converts RUNOFF input text to TEX input text
: 104 0382 1
: 105 0383 1 FORMAL PARAMETERS:
: 106 0384 1
: 107 0385 1 RNO_LEN - Length of input text
: 108 0386 1 RNO_PTR - CH$PTR to input text
: 109 0387 1 DSC - Address of output string descriptor
: 110 0388 1
: 111 0389 1 IMPLICIT INPUTS:
: 112 0390 1
: 113 0391 1 None
: 114 0392 1
: 115 0393 1 IMPLICIT OUTPUTS:
: 116 0394 1
: 117 0395 1 None
: 118 0396 1
: 119 0397 1 ROUTINE VALUE:
: 120 0398 1 COMPLETION CODES:
: 121 0399 1
: 122 0400 1 None
: 123 0401 1
: 124 0402 1 SIDE EFFECTS:
: 125 0403 1
: 126 0404 1 None
: 127 0405 1 --
: 128 0406 2 BEGIN
: 129 0407 2
: 130 0408 2 LOCAL
: 131 0409 2 LINE : VECTOR [CH$ALLOCATION (BUFFER_SIZE + 50)],
: 132 0410 2 LEADING_BLANKS,
: 133 0411 2 I_PTR, ! Copy of input line pointer
: 134 0412 2 I_LEN, ! Copy of input line length
: 135 0413 2 L_PTR, ! Pointer to output line
: 136 0414 2 L_LEN, ! Length of output line
: 137 0415 2 CURRENT_FONT, ! Font currently in use
: 138 0416 2 NEXT_FONT, ! Font of next character
: 139 0417 2 CH; ! Current character
: 140 0418 2
: 141 0419 2 L_LEN = 0; ! No characters in output string
: 142 0420 2 L_PTR = CH$PTR (LINE); ! Point to beginning of output string
: 143 0421 2 CURRENT_FONT = NORMAL_FONT; ! Font currently in use
: 144 0422 2 NEXT_FONT = NORMAL_FONT; ! Font of next character
: 145 0423 2
: 146 0424 2 I_LEN = .RNO_LEN;
: 147 0425 2 I_PTR = CH$PCUS (.RNO_PTR, .RNO_LEN - 1);
: 148 0426 2
: 149 0427 2 DECR I FROM .RNO_LEN - 1 TO 0 DO
: 150 0428 2 BEGIN
: 151 0429 2
: 152 0430 2 ! Remove trailing whitespace
: 153 0431 2
```

```
154      0432      CH = CH$RCHAR (.I_PTR);
155      0433
156      0434      IF (.CH NEQ %C' ') AND (.CH NEQ TAB) THEN EXITLOOP;
157      0435
158      0436      I_LEN = .I;
159      0437      I_PTR = CH$PLUS (.I_PTR, -1);
160      0438      END;
161      0439
162      0440      I_PTR = .RNO_PTR;
163      0441      LEADING_BLANKS = 0;
164      0442
165      0443      DECR I FROM .I_LEN TO 1 DO
166      0444      |
167      0445      | Count the number of leading blanks
168      0446      |
169      0447      | IF CH$RCHAR_A (I_PTR) EQL %C' '
170      0448      | THEN
171      0449      | LEADING_BLANKS = .LEADING_BLANKS + 1
172      0450      | ELSE
173      0451      | BEGIN
174      0452      | I_LEN = .I;
175      0453      | I_PTR = CH$PLUS (.I_PTR, -1);      ! Back up to account for overshoot
176      0454      | EXITLOOP;
177      0455      | END;
178      0456
179      0457      IF .LEADING_BLANKS
180      0458      THEN
181      0459      BEGIN
182      0460      |
183      0461      | Odd number of leading blanks
184      0462      |
185      0463      | I_LEN = .I_LEN + 1;
186      0464      | I_PTR = CH$PLUS (.I_PTR, -1);
187      0465      |
188      0466      | LEADING_BLANKS = .LEADING_BLANKS - 1;
189      0467      | END;
190      0468
191      0469      INCR I FROM 1 TO .LEADING_BLANKS / 2 DO
192      0470      BEGIN
193      0471      |
194      0472      | Insert indent sequence for each pair of leading blanks
195      0473      |
196      0474      | CH$MOVE (8, CH$PTR (UPLIT ('\mspace ')), .L_PTR);
197      0475      | L_PTR = CH$PLUS (.L_PTR, 8);
198      0476      | L_LEN = .L_LEN + 8;
199      0477      | END;
200      0478
201      0479      |
202      0480      | Process text portion of line
203      0481      |
204      0482      INCR I FROM 1 TO .I_LEN DO
205      0483      BEGIN
206      0484      IF .L_LEN GEQ BUFFER_SIZE
207      0485      THEN
208      0486      BEGIN
209      0487      |
210      0488      | String too long for buffer
```



```
211      0489 4 !
212      L 0490 4 %IF %BLISS (BLISS32)
213      0491 4 %THEN
214      0492 4 ! Signal errors in BLISS32
215      0493 4 SIGNAL (INDEX$_TRUNCATED, 0, INDEX$_TEXTD, 2, .L_LEN, CH$PTR (LINE));
216      0494 4
217      U 0495 4 %ELSE
218      0496 4 ! Use $XPO_PUT_MSG otherwise
219      0497 4 $XPO_PUT_MSG (SEVERITY = WARNING,
220      0498 4 STRING = 'string too long - truncated',
221      0499 4 STRING = $STR_CONCAT ('entry text: ', (.L_LEN, CH$PTR (LINE))));
222      U 0500 4
223      0501 4 %FI
224      0502 4
225      0503 4 EXITLOOP;
226      0504 4 END;
227      0505 4
228      0506 4 CH = CH$RCHAR_A (I_PTR);
229      0507 4 ! Get next character
230      0508 4
231      0509 4 SELECTONE .CH OF
232      0510 4 SET
233      0511 4 [%C'*']:
234      0512 4 |
235      0513 4 | Bold next character
236      0514 4 |
237      0515 4 | NEXT_FONT = .NEXT_FONT OR BOLD_FONT;
238      0516 4 |
239      0517 4 [%C'&']:
240      0518 4 |
241      0519 4 | Next character is italic
242      0520 4 |
243      0521 4 | NEXT_FONT = .NEXT_FONT OR ITALIC_FONT;
244      0522 4 |
245      0523 4 [%C'%']:
246      0524 4 | BEGIN
247      0525 4 |
248      0526 4 | Overstrike previous character - ignored
249      0527 4 |
250      L 0528 4 %IF %BLISS (BLISS32)
251      0529 4 %THEN
252      0530 4 ! Signal errors for BLISS32
253      0531 4 SIGNAL (INDEX$_OVERSTRK, 0, INDEX$_TEXTD, 2, .RNO_LEN, .RNO_PTR);
254      0532 4
255      U 0533 4 %ELSE
256      0534 4 ! Use $XPO_PUT_MSG otherwise
257      0535 4 $XPO_PUT_MSG (SEVERITY = WARNING,
258      0536 4 STRING = 'The following line contains an overstrike sequence',
259      0537 4 STRING = $STR_CONCAT ('entry text: ', (.RNO_LEN, .RNO_PTR)));
260      U 0538 4
261      0539 4 %FI
262      0540 4
263      0541 4 CH$RCHAR_A (I_PTR);
264      0542 4 ! Ignore next character
265      0543 4 I = .I + 1;
266      0544 4 END;
267      0545 4 [OTHERWISE]:
```

```
268 0546 4 BEGIN
269 0547 4
270 0548 4 A 'normal' character
271 0549 4
272 0550 4 IF .CH EQL %C'_'
273 0551 4 THEN
274 0552 5 BEGIN
275 0553 5
276 0554 5 Character is quoted.
277 0555 5 Get character.
278 0556 5
279 0557 5 CH = CH$RCHAR_A (I_PTR);
280 0558 5 I = .I + 1;
281 0559 4 END;
282 0560 4
283 0561 5 IF (.CURRENT_FONT NEQ .NEXT_FONT)
284 0562 5 AND (.CH NEQ %C' ')
285 0563 4 THEN
286 0564 5 BEGIN
287 0565 5
288 0566 5 Next character is not of the same font and is non-blank.
289 0567 5
290 0568 5 IF .CURRENT_FONT NEQ NORMAL_FONT
291 0569 5 THEN
292 0570 6 BEGIN
293 0571 6
294 0572 6 Turn off old font.
295 0573 6
296 0574 6 CH$WCHAR_A (%C')', L_PTR);
297 0575 6 L_LEN = .L_LEN + 1;
298 0576 5 END;
299 0577 5
300 0578 5 IF .NEXT_FONT NEQ NORMAL_FONT
301 0579 5 THEN
302 0580 6 BEGIN
303 0581 6
304 0582 6 Turn on new font.
305 0583 6
306 0584 6 CH$WCHAR_A (%C'(', L_PTR);
307 0585 6 CH$WCHAR_A (%C'\', L_PTR);
308 0586 6
309 0587 6 SELECTONE .NEXT_FONT OF
310 0588 6 SET
311 0589 6
312 0590 6 [BOLD_FONT]:
313 0591 7 BEGIN
314 0592 7 CH$WCHAR_A (%C'b', L_PTR);
315 0593 7 CH$WCHAR_A (%C'f', L_PTR);
316 0594 6 END;
317 0595 6
318 0596 6 [ITALIC_FONT]:
319 0597 7 BEGIN
320 0598 7 CH$WCHAR_A (%C'i', L_PTR);
321 0599 7 CH$WCHAR_A (%C't', L_PTR);
322 0600 6 END;
323 0601 6
324 0602 6 [BOLD_ITALIC]:
```



```

325      0603 7      BEGIN
326      0604 7      CH$WCHAR_A (%C'b', L_PTR);
327      0605 7      CH$WCHAR_A (%C'i', L_PTR);
328      0606 6      END;
329      0607 6
330      0608 6      TES;
331      0609 6
332      0610 6      CH$WCHAR_A (%C' ', L_PTR);
333      0611 6      L_LEN = L_LEN + 5;
334      0612 5      END;
335      0613 5
336      0614 5      CURRENT_FONT = .NEXT_FONT;
337      0615 4      END;
338      0616 4
339      0617 4      NEXT_FONT = NORMAL_FONT;
340      0618 4
341      0619 5      IF (.CH LSS %C' ') OR (.CH GTR %O'176')
342      0620 4      THEN
343      0621 5      BEGIN
344      0622 5      ! Character is a control character which are not allowed
345      0623 5
346      0624 5      ! IF %BLISS (BLISS32)
347      L 0625 5      ! Signal errors in BLISS32
348      0626 5      ! THEN
349      0627 5
350      0628 5      SIGNAL (INDEX$_CTRLCHAR, 0, INDEX$_TEXTD, 2, .RNO_LEN, .RNO_PTR);
351      0629 5
352      0630 5      ! ELSE
353      0631 5
354      0632 5      $XPO_PUT MSG (SEVERITY = WARNING,
355      0633 5      STRING = 'the following line contains control characters - ignored',
356      0634 5      STRING = $STR_CONCAT ('entry text: ', (.RNO_LEN, .RNO_PTR)));
357      0635 5
358      0636 5      ! FI
359      0637 5
360      0638 5      END
361      0639 4      ELSE
362      0640 5      BEGIN
363      0641 5
364      0642 5      SELECTONE .CH OF
365      0643 5      SET
366      0644 5
367      0645 5      [%C' ']:
368      0646 6      BEGIN
369      0647 6      CH$MOVE (4, CH$PTR (UPLIT ('{\_}')), .L_PTR);
370      0648 6      L_PTR = CH$PLUS (.L_PTR, 4);
371      0649 6      L_LEN = L_LEN + 4;
372      0650 5      END;
373      0651 5
374      0652 5      [%C'#']:
375      0653 6      BEGIN
376      0654 6      CH$MOVE (4, CH$PTR (UPLIT ('{\#}')), .L_PTR);
377      0655 6      L_PTR = CH$PLUS (.L_PTR, 4);
378      0656 6      L_LEN = L_LEN + 4;
379      0657 5      END;
380      0658 5
381      0659 5      [%C'$']:
```

```
382 0660 6 BEGIN
383 0661 6 CH$MOVE (4, CH$PTR (UPLIT ('(\$)')), .L_PTR);
384 0662 6 L_PTR = CH$PLUS (.L_PTR, 4);
385 0663 6 L_LEN = .L_LEN + 4;
386 0664 5 END;
387 0665 5
388 0666 5
389 0667 6
390 0668 6
391 0669 6 CH$MOVE (4, CH$PTR (UPLIT ('(\%)')), .L_PTR);
392 0670 6 L_PTR = CH$PLUS (.L_PTR, 4);
393 0671 6 L_LEN = .L_LEN + 4;
394 0672 5 END;
395 0673 5
396 0674 6
397 0675 6
398 0676 6 CH$MOVE (4, CH$PTR (UPLIT ('(\&)')), .L_PTR);
399 0677 6 L_PTR = CH$PLUS (.L_PTR, 4);
400 0678 5 L_LEN = .L_LEN + 4;
401 0679 5 END;
402 0680 5
403 0681 6
404 0682 6 CH$MOVE (9, CH$PTR (UPLIT ('(\rslash)')), .L_PTR);
405 0683 6 L_PTR = CH$PLUS (.L_PTR, 9);
406 0684 6 L_LEN = .L_LEN + 9;
407 0685 5 END;
408 0686 5
409 0687 5
410 0688 6
411 0689 6
412 0690 6 CH$MOVE (4, CH$PTR (UPLIT ('(\@)')), .L_PTR);
413 0691 6 L_PTR = CH$PLUS (.L_PTR, 4);
414 0692 5 L_LEN = .L_LEN + 4;
415 0693 5 END;
416 0694 5
417 0695 6
418 0696 6
419 0697 6 CH$MOVE (4, CH$PTR (UPLIT ('(\^)')), .L_PTR);
420 0698 6 L_PTR = CH$PLUS (.L_PTR, 4);
421 0699 5 L_LEN = .L_LEN + 4;
422 0700 5 END;
423 0701 5
424 0702 6
425 0703 6 CH$WCHAR_A (XC'\', L_PTR);
426 0704 6 CH$WCHAR_A (XC'{', L_PTR);
427 0705 6 L_LEN = .L_LEN + 2;
428 0706 5 END;
429 0707 5
430 0708 5
431 0709 6
432 0710 6 CH$WCHAR_A (XC'\', L_PTR);
433 0711 6 CH$WCHAR_A (XC'}', L_PTR);
434 0712 6 L_LEN = .L_LEN + 2;
435 0713 5 END;
436 0714 5
437 0715 5
438 0716 6 [OTHERWISE]:
BEGIN
```



```
439 0717 6
440 0718 6
441 0719 6
442 0720 6
443 0721 6
444 0722 5
445 0723 5
446 0724 5
447 0725 5
448 0726 4
449 0727 4
450 0728 3
451 0729 3
452 0730 3
453 0731 2
454 0732 2
455 0733 2
456 0734 2
457 0735 2
458 0736 2
459 0737 2
460 0738 2
461 0739 2
462 0740 3
463 0741 3
464 0742 2
465 0743 2
466 0744 2
467 0745 2
468 0746 2
469 0747 2
470 0748 1
```

```

      |
      | A real normal character
      |
CH$WCHAR_A (.CH, L_PTR);
L_LEN = .L_LEN + 1;
END;

      TES;

      END;

      END;

      TES;

      END;

IF .CURRENT_FONT NEQ NORMAL_FONT
THEN
  BEGIN
    | Turn off bold and italics
    |
    CH$WCHAR_A (%C')', L_PTR);
    L_LEN = .L_LEN + 1;
    END;

    |
    | Copy line to output string
    |
    $STR_COPY (STRING = (.L_LEN, CH$PTR (LINE)), TARGET = .DSC);
  END;
```

```

      .TITLE NDXTEX NDXTEX -- RUNOFF to TEX conversion
      .IDENT \V04-000\
      .PSECT $SPLIT$,NOWRT,NOEXE,2

20 65 63 61 70 73 6D 5C 00000 P.AAA: .ASCII <92>\mspace \
7D 5F 5C 7B 00008 P.AAB: .ASCII \{\<92>\} \
7D 23 5C 7B 0000C P.AAC: .ASCII \{\<92>\#} \
7D 24 5C 7B 00010 P.AAD: .ASCII \{\<92>\$} \
7D 25 5C 7B 00014 P.AAE: .ASCII \{\<92>\%} \
7D 26 5C 7B 00018 P.AAF: .ASCII \{\<92>\&} \
00 00 00 7D 68 73 61 6C 73 72 5C 7B 0001C P.AAG: .ASCII \{\<92>\rslash}\<0><0><0>
7D 40 5C 7B 00028 P.AAH: .ASCII \{\<92>\@} \
7D 5E 5C 7B 0002C P.AAI: .ASCII \{\<92>\^} \

      .EXTRN DSRINDEX$_BADLOGIC
      .EXTRN DSRINDEX$_BADVALUE
      .EXTRN DSRINDEX$_INSVIRMEM
      .EXTRN DSRINDEX$_LINELENG
      .EXTRN DSRINDEX$_NOREF
      .EXTRN DSRINDEX$_OPENIN
      .EXTRN DSRINDEX$_OPENOUT
      .EXTRN DSRINDEX$_TOOMANY
```

```
.EXTRN DSRINDEX$_VALERR
.EXTRN DSRINDEX$_CANTBAL
.EXTRN DSRINDEX$_CLOSEQUOT
.EXTRN DSRINDEX$_CONFQUAL
.EXTRN DSRINDEX$_CTRLCHAR
.EXTRN DSRINDEX$_DOESNTFIT
.EXTRN DSRINDEX$_DUPBEGIN
.EXTRN DSRINDEX$_EMPTYIN
.EXTRN DSRINDEX$_IGNORED
.EXTRN DSRINDEX$_INVINPUT
.EXTRN DSRINDEX$_INVRECORD
.EXTRN DSRINDEX$_LASTCONT
.EXTRN DSRINDEX$_NOBEGIN
.EXTRN DSRINDEX$_NOEND
.EXTRN DSRINDEX$_NOINDEX
.EXTRN DSRINDEX$_NOLIST
.EXTRN DSRINDEX$_OVERSTRK
.EXTRN DSRINDEX$_SKIPPED
.EXTRN DSRINDEX$_SYNTAX
.EXTRN DSRINDEX$_TEXTFILE
.EXTRN DSRINDEX$_TOODEEP
.EXTRN DSRINDEX$_TOOFEW
.EXTRN DSRINDEX$_TRUNCATED
.EXTRN DSRINDEX$_COMPLETE
.EXTRN DSRINDEX$_CREATED
.EXTRN DSRINDEX$_IDENT
.EXTRN DSRINDEX$_PROFILE
.EXTRN DSRINDEX$_TEXT, DSRINDEX$_TEXTD
.EXTRN DSRINDEX$_TMS11
.EXTRN TAB, XST$COPY, STR$FAILURE

.PSECT $CODE$,NOWRT,2

.OFFC 00000
.ENTRY RNOTEX, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,- R11
MOVAB -2068(SP), SP
MOVAB LINE, L_PTR
CLRL CURRENT_FONT
CLRL NEXT_FONT
MOVL RNO_LEN, I_LEN
ADDL3 RNO_LEN, RNO_PTR, R6
DECL I_PTR
MOVL RNO_LEN, I
BRB 3$
MOVZBL (I_PTR), CH
CPL CH, #32
BEQL 2$
CPL CH, #TAB
BNEQ 4$
MOVL I, I_LEN
DECL I_PTR
SOBGEQ I, 1$
MOVL RNO_PTR, I_PTR
CLRL LEADING_BLANKS
MOVAB 1(R7), I
BRB 7$
MOVZBL (I_PTR)+, R2
```

5E	F7EC	CE	9E	00002	
58	10	AE	9E	00007	
		7E	D4	0000B	
		5A	7C	0000D	
56	08	57	04	AC	D0 0000F
		AC	04	AC	C1 00013
		56		D7	00019
		50	04	AC	D0 0001B
				16	11 0001F
		59		66	9A 00021 1\$:
		20		59	D1 00024
				09	13 00027
00000000G		8F		59	D1 00029
				08	12 00030
		57		50	D0 00032 2\$:
				56	D7 00035
		E7		50	F4 00037 3\$:
		56	08	AC	D0 0003A 4\$:
				51	D4 0003E
		50	01	A7	9E 00040
				13	11 00044
		52		86	9A 00046 5\$:

	20		52	91	00049	CMPB	R2, #32		
			04	12	0004C	BNEQ	6\$		
			51	D6	0004E	INCL	LEADING_BLANKS		0449
			07	11	00050	BRB	7\$		
	57		50	D0	00052	6\$: MOVL	I, I_LEN		0452
			56	D7	00055	DECL	I_PTR		0453
			03	11	00057	BRB	8\$		0451
	EA		50	F5	00059	7\$: SOBGTR	I, 5\$		0447
	06		51	E9	0005C	8\$: BLBC	LEADING_BLANKS, 9\$		0457
			57	D6	0005F	INCL	I_LEN		0463
			56	D7	00061	DECL	I_PTR		0464
			51	D7	00063	DECL	LEADING_BLANKS		0466
08	AE	51	02	C7	00065	9\$: DIVL3	#2, LEADING_BLANKS, 8(SP)		0469
			04	AE	D4	0006A	CLRL	I	
			0E	11	0006D	BRB	11\$		
	68	00000000'	EF	08	28	0006F	10\$: MOVCL3	#8, P.AAA, (L_PTR)	0474
			58	08	C0	00077	ADDL2	#8, L_PTR	0475
			5B	08	C0	0007A	ADDL2	#8, L_LEN	0476
	EC	04	AE	08	AE	F3	0007D	11\$: AOBLEQ	8(SP), I, 10\$
			08	AE	D4	00083	CLRL	I	0482
			64	11	00086	BRB	16\$		
	000007D0	8F	5B	D1	00088	12\$: CMPL	L_LEN, #2000		0484
			1F	19	0008F	BLSS	13\$		
			14	AE	9F	00091	PUSHAB	LINE	0493
			5B	DB	00094	PUSHL	L_LEN		
			02	DD	00096	PUSHL	#2		
		00000000G	8F	DD	00098	PUSHL	#DSRINDEX\$_TEXTD		
			7E	D4	0009E	CLRL	-(SP)		
		00000000G	8F	DD	000A0	PUSHL	#DSRINDEX\$_TRUNCATED		
	00000000G	00	06	FB	000A6	CALLS	#6, LIB\$SIGNAL		
			01	92	31	000AD	BRW	42\$	0486
			59	86	9A	000B0	13\$: MOVZBL	(I_PTR)+, CH	0506
			2A	59	D1	000B3	CMPL	CH, #42	0511
			05	12	000B6	BNEQ	14\$		
			5A	01	88	000B8	BISB2	#1, NEXT_FONT	0515
			2F	11	000BB	BRB	16\$		
			26	59	D1	000BD	14\$: CMPL	CH, #38	0517
			05	12	000C0	BNEQ	15\$		
			5A	02	88	000C2	BISB2	#2, NEXT_FONT	0521
			25	11	000C5	BRB	16\$		
			25	59	D1	000C7	15\$: CMPL	CH, #37	0523
			23	12	000CA	BNEQ	17\$		
			7E	04	AC	7D	000CC	MOVQ	RNO_LEN, -(SP)
			02	DD	000D0	PUSHL	#2		0531
		00000000G	8F	DD	000D2	PUSHL	#DSRINDEX\$_TEXTD		
			7E	D4	000D8	CLRL	-(SP)		
		00000000G	8F	DD	000DA	PUSHL	#DSRINDEX\$_OVERSTRK		
	00000000G	00	06	FB	000E0	CALLS	#6, LIB\$SIGNAL		
			56	D6	000E7	INCL	I_PTR		0541
			08	AE	D6	000E9	INCL	I	0542
			01	4C	31	000EC	16\$: BRW	41\$	0508
	0000005F	8F	59	D1	000EF	17\$: CMPL	CH, #95		0550
			06	12	000F6	BNEQ	18\$		
			59	86	9A	000F8	MOVZBL	(I_PTR)+, CH	0557
			08	AE	D6	000FB	INCL	I	0558
			5A	6E	D1	000FE	18\$: CMPL	CURRENT_FONT, NEXT_FONT	0561
			4E	13	00101	BEQL	25\$		

20		59	D1	00103	CMPL	CH, #32	0562
		49	13	00106	BEQL	25\$	0568
		6E	D5	00108	TSTL	CURRENT_FONT	0574
		06	13	0010A	BEQL	19\$	0575
88	7D	8F	90	0010C	MOVB	#125, (L_PTR)+	0578
		5B	D6	00110	INCL	L_LEN	0584
		5A	D5	00112	TSTL	NEXT_FONT	0590
		38	13	00114	BEQL	24\$	0592
88	5C7B	8F	B0	00116	MOVW	#23675, (L_PTR)+	0593
01		5A	D1	0011B	CMPL	NEXT_FONT, #1	0596
		0A	12	0011E	BNEQ	20\$	0598
88	62	8F	90	00120	MOVB	#98, (L_PTR)+	0599
68	66	8F	90	00124	MOVB	#102, (L_PTR)	0602
		1C	11	00128	BRB	22\$	0604
02		5A	D1	0012A	CMPL	NEXT_FONT, #2	0605
		0A	12	0012D	BNEQ	21\$	0610
88	69	8F	90	0012F	MOVB	#105, (L_PTR)+	0611
68	74	8F	90	00133	MOVB	#116, (L_PTR)	0614
		0D	11	00137	BRB	22\$	0617
03		5A	D1	00139	CMPL	NEXT_FONT, #3	0619
		0A	12	0013C	BNEQ	23\$	0628
88	62	8F	90	0013E	MOVB	#98, (L_PTR)+	0619
68	69	8F	90	00142	MOVB	#105, (L_PTR)	0645
		58	D6	00146	INCL	L_PTR	0647
88		20	90	00148	MOVB	#32, (L_PTR)+	0648
5B		05	C0	0014B	ADDL2	#5, L_LEN	0652
6E		5A	D0	0014E	MOVL	NEXT_FONT, CURRENT_FONT	0654
		5A	D4	00151	CLRL	NEXT_FONT	0655
20		59	D1	00153	CMPL	CH, #32	0659
		09	19	00156	BLSS	26\$	0661
0000007E		8F	D1	00158	CMPL	CH, #126	0662
		1D	15	0015F	BLEQ	27\$	0666
7E	04	AC	7D	00161	MOVQ	RNO_LEN, -(SP)	0668
		02	DD	00165	PUSHL	#2	0669
	00000000G	8F	DD	00167	PUSHL	#DSRINDEX\$, TEXTD	0673
		7E	D4	0016D	CLRL	-(SP)	0674
	00000000G	8F	DD	0016F	PUSHL	#DSRINDEX\$, CTRLCHAR	0675
00000000G	00	06	FB	00175	CALLS	#6, LIB\$SIGNAL	0676
		61	11	0017C	BRB	33\$	0677
0000005F	8F	59	D1	0017E	CMPL	CH, #95	0678
		09	12	00185	BNEQ	28\$	0679
68	00000000'	EF	D0	00187	MOVL	P.AAB, (L_PTR)	0680
		73	11	0018E	BRB	36\$	0681
23		59	D1	00190	CMPL	CH, #35	0682
		09	12	00193	BNEQ	29\$	0683
68	00000000'	EF	D0	00195	MOVL	P.AAC, (L_PTR)	0684
		65	11	0019C	BRB	36\$	0685
24		59	D1	0019E	CMPL	CH, #36	0686
		09	12	001A1	BNEQ	30\$	0687
68	00000000'	EF	D0	001A3	MOVL	P.AAD, (L_PTR)	0688
		57	11	001AA	BRB	36\$	0689
25		59	D1	001AC	CMPL	CH, #37	0690
		09	12	001AF	BNEQ	31\$	0691
68	00000000'	EF	D0	001B1	MOVL	P.AAE, (L_PTR)	0692
		49	11	001B8	BRB	36\$	0693
26		59	D1	001BA	CMPL	CH, #38	0694
		09	12	001BD	BNEQ	32\$	0695

68	00000000'	EF	D0	001BF	MOVL	P.AAF, (L_PTR)	:	0675		
		3B	11	001C6	BRB	36\$	:	0676		
0000005C	8F	59	D1	001C8	32\$:	CMPL	CH, #92	:	0680	
		10	12	001CF	BNEQ	34\$	:	:	:	
68	00000000'	EF	09	28	001D1	MOV C3	#9, P.AAG, (L_PTR)	:	0682	
		58	09	C0	001D9	ADDL2	#9, L_PTR	:	0683	
		5B	09	C0	001DC	ADDL2	#9, L_LEN	:	0684	
		5A	11	001DF	33\$:	BRB	41\$	:	0642	
00000040	8F	59	D1	001E1	34\$:	CMPL	CH, #64	:	0687	
		09	12	001E8	BNEQ	35\$	:	:	:	
68	00000000'	EF	D0	001EA	MOVL	P.AAH, (L_PTR)	:	0689		
		10	11	001F1	BRB	36\$	:	0690		
0000005E	8F	59	D1	001F3	35\$:	CMPL	CH, #94	:	0694	
		0F	12	001FA	BNEQ	37\$	:	:	:	
68	00000000'	EF	D0	001FC	MOVL	P.AAI, (L_PTR)	:	0696		
		04	C0	00203	36\$:	ADDL2	#4, L_PTR	:	0697	
		04	C0	00206	ADDL2	#4, L_LEN	:	0698		
		30	11	00209	BRB	41\$	:	0642		
0000007B	8F	59	D1	0020B	37\$:	CMPL	CH, #123	:	0701	
		0A	12	00212	BNEQ	38\$	:	:	:	
	88	5C	8F	90	00214	MOVB	#92, (L_PTR)+	:	0703	
	68	7B	8F	90	00218	MOVB	#123, (L_PTR)	:	0704	
		11	11	0021C	BRB	39\$	:	:	:	
0000007D	8F	59	D1	0021E	38\$:	CMPL	CH, #125	:	0708	
		0F	12	00225	BNEQ	40\$	:	:	:	
	88	5C	8F	90	00227	MOVB	#92, (L_PTR)+	:	0710	
	68	7D	8F	90	0022B	MOVB	#125, (L_PTR)	:	0711	
		58	D6	0022F	39\$:	INCL	L_PTR	:	:	
	5B	02	C0	00231	ADDL2	#2, L_LEN	:	0712		
		05	11	00234	BRB	41\$	:	0642		
	88	59	90	00236	40\$:	MOVB	CH, (L_PTR)+	:	0720	
		5B	D6	00239	INCL	L_LEN	:	0721		
FE46	08	AE	01	57	F1	0023B	41\$:	ACBL	I_LEN, #1, I, 12\$	0482
				6E	D5	00242	42\$:	TSTL	CURRENT_FONT	0734
				06	13	00244	BEQL	43\$	:	:
	88	7D	8F	90	00246	MOVB	#125, (L_PTR)+	:	0740	
			5B	D6	0024A	INCL	L_LEN	:	0741	
	OC	AE	5B	B0	0024C	43\$:	MOVW	L_LEN, \$STR\$STRING	:	0747
	OE	AE	0E	90	00250	MOVB	#T4, \$STR\$STRING+2	:	:	:
	OF	AE	01	90	00254	MOVB	#1, \$STR\$STRING+3	:	:	:
	10	AE	AE	9E	00258	MOVAB	LINE, \$STR\$STRING+4	:	:	:
			00000000G	EF	9F	0025D	PUSHAB	STR\$FAILURE	:	:
				7E	D4	00263	CLRL	-(SP)	:	:
		OC	AC	DD	00265	PUSHL	DSC	:	:	:
		18	AE	9F	00268	PUSHAB	\$STR\$STRING	:	:	:
			7E	D4	0026B	CLRL	-(SP)	:	:	:
00000000G	EF	05	FB	0026D	CALLS	#5, XST\$COPY	:	:	:	:
			04	00274	RET		:	0748	:	:

; Routine Size: 629 bytes, Routine Base: \$CODE\$ + 0000

: 471 0749 1
: 472 0750 1 END
: 473 0751 0 ELUDOM

! End of module

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
\$PLITS	48 NOVEC,NOWRT, RD ,NOEXE,NOSHR,	LCL, REL, CON,NOPIC,ALIGN(2)
\$CODE\$	629 NOVEC,NOWRT, RD , EXE,NOSHR,	LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]XPORT.L32;1	590	20	3	252	00:00.2

COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:NDXTEX/OBJ=OBJ\$:NDXTEX MSRC\$:NDXTEX/UPDATE=(ENH\$:NDXTEX)

; Size: 629 code + 48 data bytes
; Run Time: 00:16.1
; Elapsed Time: 00:33.5
; Lines/CPU Min: 2795
; Lexemes/CPU-Min: 25704
; Memory Used: 265 pages
; Compilation Complete

0345 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY